tri.py

someonesdad1@gmail.com 25 Jan 2013

Occasionally we need to solve triangles. While the usual approach is with paper, pencil, and calculator, a computer program can do it faster -- and this provides a convenience if you have a number of similar problems to solve.

The `tri.zip` package from http://code.google.com/p/hobbyutil/ contains the `tri.py` python script, the `sig.py` module which does formatting of numbers, and `tri.pdf`, this document. One advantage of the `tri.py` script is that you can enter uncertainties for the sides and angles, then see the uncertainties in the calculated results.

You'll need to have the python programming language installed on your computer. You can get python from http://www.python.org/. If you want to use the uncertainties feature, you'll need to download and install the python `uncertainties` module from http://pypi.python.org/pypi/uncertainties/. The `uncertainties` module is optional and the script will work without it. The `tri.py` script was tested with python 2.6.5 and 2.7.2. **Note it does not work correctly with python 3**. If you get it working with python 3, please send me a patch and I'll fix the script (I don't use python 3 because I need too many older libraries in my work).

Install the `tri.py` and `sig.py` scripts from the `tri.zip` package in a convenient directory. If you want to use the `tri.py` script from a directory different from where `sig.py` is, you'll need to put `sig.py`'s directory in your `PYTHONPATH` environment variable.

Usage

To get a manpage printed to stdout, invoke the `tri.py` script with the `-h` option.

There are two ways to use the `tri.py` script. If you start it with no command line parameter, you'll be prompted for the type of problem you want to solve, the relevant sides and angles of the triangle, and whether you want to work in degrees or radian measure for angles.

The other way to use the script is to include a datafile on the command line. The datafile is a text file that contains the same information you're prompted for in interactive mode. There must be one line for each entry. Entries are of two types: control words and assignments.

A control word is one of the following:

`deg`	Sets the angle mode to degrees. Using degrees is the default.
`rad`	Sets the angle mode to radians.
`sss`	Solve a side-side-side problem. sss is the default.
`ssa`	Solve a side-side-angle problem.
`sas`	Solve a side-angle-side problem.
`saa`	Solve a side-angle-angle problem.
`asa`	Solve an angle-side-angle problem.

Assignments assign a value to a variable. You can use any valid python identifier as a name and the value can be an expression. Thus, for example, the following is a valid assignment

```
unc = sqrt(0.1**2 + 0.075**2)
```

The `math` module's symbols are in scope, so you can use math functions like `sqrt`, `sin`, etc.

For the sides and angles of the triangle, you must use `S1`, `S2`, and `S3` for the sides and `A1`, `A2`, and `A3` for the angles. For example, if the problem requires two angles, you'd use `A1` and `A2`. The script will stop with an error message if it doesn't have the correct variables for a problem.

A twist is that these assignments can have an associated uncertainty. This is indicated by a number or expression in square brackets. For example, to assign an uncertainty to side 1 of 0.1, you could

use a line like:

```
S1 = 122 [0.1]
```

Note: when you are being prompted in interactive mode for numbers, you can enter an assignment statement and it will be accepted if it is valid; you'll be re-prompted for the needed side or angle. Also note you can type in numbers like `100 [0.5]` in interactive mode and they will be interpreted as a number with an uncertainty, just as in the non-interactive fashion.

Examples

For the following examples, I'll use datafiles, but you can type the data in manually if you want to use interactive mode.

Simple problem

Let's start with a sss problem where the three sides are equal to 100 (an equilateral triangle). The following lines are put into a file named `datafile`:

```
sss
S1 = 100
S2 = 100
S3 = 100
```

When the script is run by the command `python tri.py datafile`, the results are

```
Triangle solution:
  Sides                 100.0           100.0           100.0
  Angles (deg)          60.00           60.00           60.00
  Area                  4330
  Perimeter             300.0
  Inscribed circle      radius = 28.87, diameter = 57.74
  Circumscribed circle  radius = 57.74, diameter = 115.5
```

The results are as we'd expect. The area is $0.5(100)^2\cos(60\,°)$ or $5000\sqrt{3}/2$.

Suppose we had measured these sides of the triangle with a rule and we felt the uncertainty of each measured side was 0.5. We can see the effect of these uncertainties with the following datafile:

```
sss
e = 0.5
S1 = 100 [e]
S2 = 100 [e]
S3 = 100 [e]
```

When this datafile is run, the results are

```
Triangle solution:
  Sides                 100.0(5)        100.0(5)        100.0(5)
  Angles (deg)          60.0(4)         60.0(4)         60.0(4)
  Area                  4330(20)
  Perimeter             300.0(9)
  Inscribed circle      radius = 28.87(8), diameter = 57.7(2)
  Circumscribed circle  radius = 57.7(2), diameter = 115.5(3)
```

The uncertainties are given in the usual short-hand fashion: 28.87(8) means the value is 28.87 and its uncertainty is 0.08.

Yard problem

Suppose I've done some simple surveying of a large triangle in my yard with a tape measure and I want to calculate the area and angles of the triangle. Because the measurements are over grass or somewhat rough ground, they don't repeat all that well. Suppose I've measured two sides in meters and the included angle in degrees. I put my measurements into a datafile:

```
sas
deg
S1 = 27.23 [0.07] # In m
```

```
S2 = 18.14 [0.1]  # In m
A1 = 37.2 [0.5]   # In degrees
```

This results in

```
Triangle solution:
  Sides                 27.23(7)        18.1(1)         16.8(2)
  Angles (deg)          102.2(7)        40.6(4)         37.2(5)
  Area                  149(2)
  Perimeter             62.2(2)
  Inscribed circle      radius = 4.80(5), diameter = 9.60(1)
  Circumscribed circle  radius = 13.93(6), diameter = 27.9(1)
```

Thus, I find that the area uncertainty is about 2 square meters.